Introduction to Assembly Language

Benjamin Kong — 1573684
Lora Ma ———— 1570935

ECE 212 Lab Section H11

February 21, 2020

Contents

1 Introduction

Assembly language is a low-level programming language that is converted to machine code using an assembler. Assembly language uses *mnemonics* to represent low-level machine instructions, and this makes it much more readable than machine code. It is important to note that assembly language is specific to a particular computer architecture and hence may or may not work on different systems.

The purpose of this lab was to become more familiar with assembly language using the NetBurner ColdFire microcontroller board. In order to gain experience with assembly language programming, two different programs were created using assembly. In the first part of the lab (part A), we created a program that converted an Ascii character to its hexadecimal equivalent. For example, '5' is converted to '5' and 'B' is converted to '11.' However, if a character without a hexadecimal equivalent is entered, such as 'J,' then an error code results.

For the second part of the lab (part B), we created a program that converts an Ascii letter into its uppercase or lowercase equivalent. For example, 'a' is converted to 'A' while 'E' is converted to 'e.' If an invalid Ascii character is input (such as an ampersand '&,' for example), an error code results.

2 Design

2.1 Part A

First, in our program we initialize address registers a2 and a3 with addresses and store the number of times the program will loop in a data register. Then, we check if at any point in our program the enter key is pressed. If it is pressed, the program exits. Otherwise, we check if our character is valid through a series of tests. First, we check if the given value is less than the hex value '0'. If it is, then it is an invalid character and the character is replaced with an error code. If the value is greater than the hex value '0', then we compare the value to the hex value '9'. If the value is greater than '9', we further test to see if the value is a character between the hex value 'A' and the hex value 'F'. If the value is smaller than the hex value 'A', then it is an invalid character. If the value is greater than F, we check if the character is between the hex value 'a' and 'f'. If the value is less than the hex value 'a' or greater than the hex value 'f', it is an invalid character. After we've established whether or not the character is valid, we convert the value to it's hexadecimal equivalent.

2.2 Part B

First, we initialize address registers a2 and a3 with addresses and store the number of times the program will loop in a data register. Then, we check if at any point in our program the enter key is pressed. If it is pressed, the program exits. Otherwise, we check if our character is valid through a series of tests. First, we check if the character is between the hex value 'A' and the hex value 'Z'. If it is less than 'A', then it is not a valid character. If it is greater than 'Z', we further test to see if the character is between the hex value 'a' and the hex value 'z'. If the character is either less than 'a' or greater than 'z', it is an invalid character. If we have an invalid character, the character is replaced with an error code and we move on to the next value. If we have a valid character, we convert the letter to it's lowercase/uppercase equivalent.